COL 380 : A2

Sidharth Agarwal - 2019CS50661

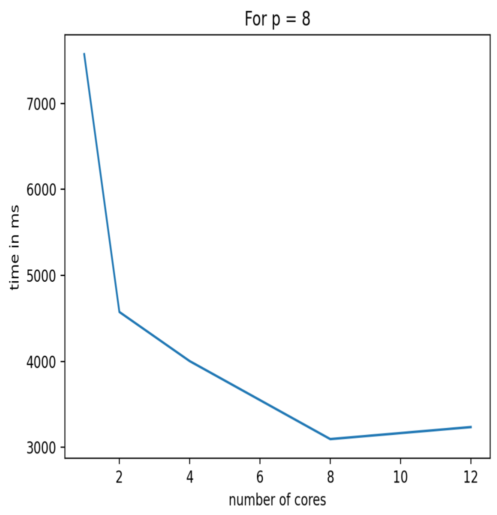
February 4, 2022

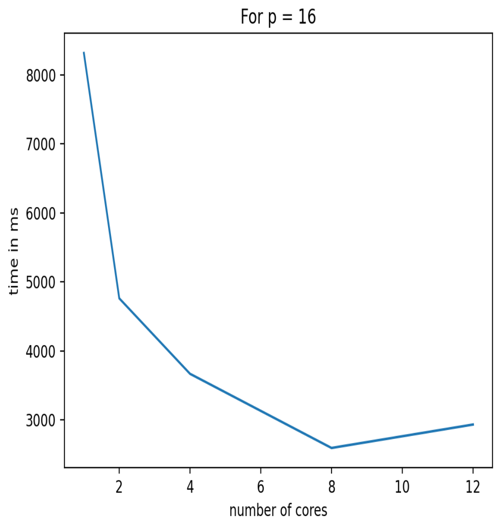
Approach

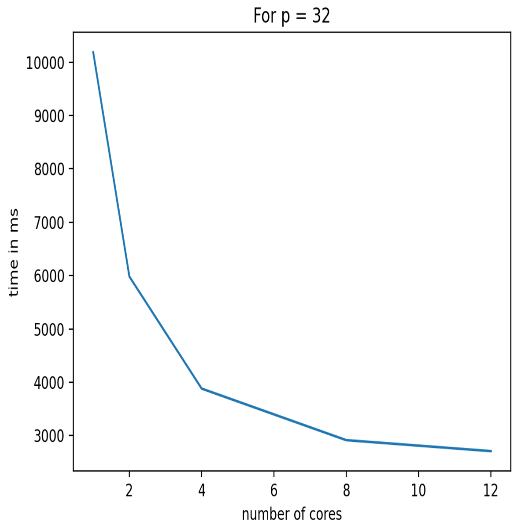
So I followed the implementation mentioned in the document. My idea was to assign each element of dataset the index of the bin to which it belongs, and keep track of bin size as well. So that I will use prefix sum over bin size and later reiterate over each element of data and put them into the right bin, at appropriate offset based on the previous elements pushed. Now for assigning each element this correct index I used tasks. Basically I created p(number of bins) tasks and in each task, I iterated over all the elements of data and checked whether this element belongs to this bin in O(1), hence parallely reducing a total computation of O(p*n) to O(n). Now I have the index of bin for each element. So I will create a dummy array and iterate over all the elements of original and putting them at appropriate place in this new dummy array. After that I copied this back to original array and called recursively ParallelSort or sequential sort(based on threshold) in p(number of bins) number of tasks. Also for the sequential sort algorithm I used mergesort.

Experiments:

So because I was not able to get my job completed on hpc, because of long queues. I am posting here results from my pc which has 4 cores. For a file of size 2^{24}, for different values of p, i.e different partitions.







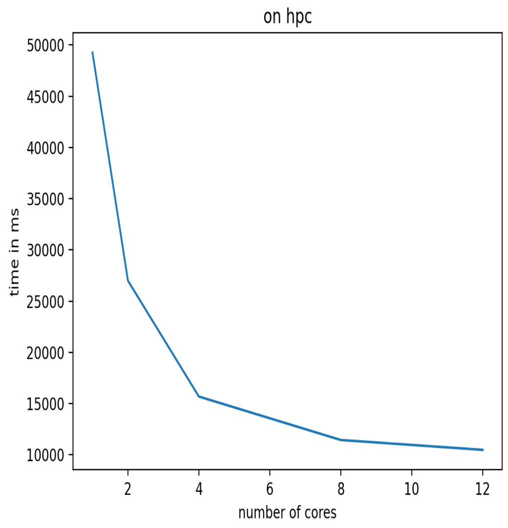
It can be observed that the number of bins donot effect the performance that much. And on a machine of 4 cores I was able to reduce time more than 3x with 8 threads. to run the program simply use

- **make**
- **make driver && ./out input_file_name num_threads**

HPC Experiments:

The data stated below has been observed over n = 2^{26} for p = 40(different value of p didn't affect that much) over upto 12 cores. I tested over number of threads = number of cores.

Threads	Time(in milliseconds)
1	49238.8
2	26991.1
4	15682.8
8	11444.8
12	10482.3



So a boost of around 5x was observed from 1 thread to 12 threads.

Failed attempt

So I also tested some more approaches one of them being, to assign the work of putting each data element to correct bin to a new task. Basically I will iterate over all the elements of the data array and for each element, I will create a task of assigning to the correct bin and increasing the size of the bin for that thread(to avoid race condition) accordingly and moving forward to assign the job for next element to another task. Later I will simply add all the bin sizes per thread to get the final bin size and then using the previous method, I will use these pointers and bin sizes to sort the array. The reason why this gives worse performance than sequential approach is because of n tasks created for the program, which brings a lot of overhead.